[19]中华人民共和国国家知识产权局

[51]Int. Cl⁷
G06F 9/48
H04L 9/00

[12]发明专利申请公开说明书

[21]申请号 01103000.3

[43]公开日 2001年8月22日

[11]公开号 CN 1309351A

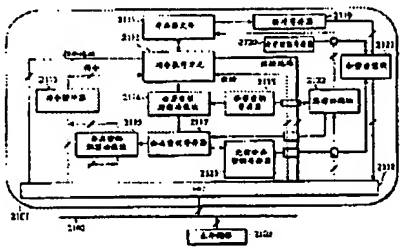
[22]申请日 2001.2.14 [21]申请号 01103000.3
[30]优先权
[32]2000.2.14 [33]JP [31]035898/2000
[32]2000.5.8 [33]JP [31]135010/2000
[71]申请人 株式会社东芝
地址 日本神奈川县
[72]发明人 桥本干生 寺本圭一 齐藤健
白川健治 藤本谦作
[74]专利代理机构 中国国际贸易促进委员会专利商标事务所
代理人 吴丽丽

权利要求书4页 说明书43页 附图页数15页

[54]发明名称 抗干预微处理器

[57]摘要

在多任务环境下，抗干预微处理器保存一个其执行被中断的程序 的上下文信息，其中该上下文信息含有指明该程序的执行状态和该程 序的执行码密钥的信息。通过从保存的上下文信息恢复该程序的执行 状态，可以重新启动该程序的执行。利用微处理器的公开密钥可以将 此上下文信息加密，然后利用微处理器的秘密密钥进行解密。



ISSN1008-4274

知识产权出版社出版

01.02.14

权 利 要 求 书

1. 一种具有不能被读出到外部的唯一秘密密钥和与该唯一秘密密钥对应的唯一公开密钥的微处理器，该微处理器包括：

读取单元，被进行配置以从外部存储器读出多个利用不同执行码密钥加密的程序；

解密单元，被进行配置以利用各自解密密钥，对多个通过读取单元读出的程序进行解密；

执行单元，被进行配置以执行多个利用解密单元解密的程序；

上下文信息保存单元，被进行配置以将其执行被中断的一个程序的上下文信息保存到外部存储器或保存到在微处理器内部设置的上下文信息存储器，该上下文信息含有指明此程序的执行状态和此程序的执行码密钥的信息；以及

重新启动单元，被进行配置以通过从外部存储器或上下文信息存储器读出上下文信息并通过从上下文信息中恢复此程序的执行状态，重新启动执行此程序。

2. 根据权利要求 1 所述的微处理器，其中所配置的上下文信息保存单元利用公开密钥对上下文信息进行加密，并将加密上下文信息保存到外部存储器；以及

所配置的重新启动单元通过从外部存储器读出加密上下文信息，利用秘密密钥解密加密上下文信息，以及从解密上下文信息中恢复一个程序的执行状态，重新启动此程序的执行。

3. 根据权利要求 2 所述的微处理器，其中仅当包含在解密上下文信息内的解密执行码密钥与此程序的执行码密钥一致时，重新启动单元才重新启动此程序的执行。

4. 根据权利要求 2 所述的微处理器，其中重新启动单元将包含在解密上下文信息内的解密执行码密钥用作解密密钥以解密此程序。

5. 根据权利要求 1 所述的微处理器，其中所配置的上下文信息保存单元以明文形式将上下文信息保存到此程序被中断后所执行的另一

01-02-14

个程序不可读的上下文信息存储器；以及

通过从上下文信息存储器读出上下文信息并从上下文信息恢复此程序的执行码，所配置的重新启动单元重新启动此程序的执行。

6. 根据权利要求 5 所述的微处理器，其中重新启动单元根据另一个程序规定的指令重新启动此程序的执行。

7. 根据权利要求 6 所述的微处理器，其中在此程序的执行被中断时，上下文信息保存单元将上下文信息保存到上下文信息存储器，并利用公开密钥将上下文信息存储器内的上下文信息加密，然后根据另一个程序规定的另一条指令的执行，将加密上下文信息存储到外部存储器。

8. 根据权利要求 5 所述的微处理器，其中在此程序的执行被中断时，上下文信息保存单元将上下文信息保存到上下文信息存储器，利用公开密钥将上下文信息存储器内的上下文信息加密，然后将加密上下文信息存储到另一个程序规定的外部存储器。

9. 根据权利要求 1 所述的微处理器，其中所配置的上下文信息保存单元产生作为临时密钥的随机数、加密上下文信息、然后将加密上下文信息存储到外部存储器，加密上下文信息含有：第一数值，通过对信息进行加密获得，利用临时密钥指明此程序的执行状态；以及第二数值，通过利用公开密钥加密临时密钥获得；以及

通过从外部存储器读出加密上下文信息，利用秘密密钥由包含在加密上下文信息内的第二数值解密获得临时密钥，利用解密的临时密钥由包含在加密上下文信息内第一数值解密出指明执行状态的信息，以及从解密上下文信息恢复此程序的执行状态，所配置的重新启动单元重新启动此程序的执行。

10. 根据权利要求 9 所述的微处理器，其中上下文信息保存单元保存还含有利用此程序的执行码密钥对临时密钥进行加密获得的第三数值的加密上下文信息。

11. 根据权利要求 10 所述的微处理器，其中重新启动单元利用秘密密钥由包含在加密上下文信息内的第二数值解密获得第一临时密

01.02.14

钥，并利用第一解密临时密钥由包含在加密上下文信息内的第一数值解密获得指明执行状态的信息，同时利用该程序的执行码密钥由包含在加密上下文信息内的第三数值解密获得第二临时密钥，然后只在第一解密的临时密钥与第二解密的临时密钥一致时，重新启动此程序的执行。

12. 根据权利要求 1 所述的微处理器，该微处理器进一步包括：

执行状态存储单元，用于存储当前执行程序的执行状态；以及

执行状态初始化单元，被进行配置以在此程序被中断后而在另一个程序开始之前，将执行状态存储单元的内容初始化为规定数值或将执行状态存储单元的内容加密。

13. 根据权利要求 1 所述的微处理器，该微处理器进一步包括：

密钥读取单元，被进行配置以从外部存储器读出被事先利用公开密钥加密的各程序的执行码密钥；以及

密钥解密单元，被进行配置以利用秘密密钥解密通过密钥读取单元读出的执行码密钥；

其中解密单元利用作为解密密钥的执行码密钥解密各程序。

14. 根据权利要求 1 所述的微处理器，该微处理器进一步包括：

执行状态存储单元，用于存储当前执行程序的执行状态和将被当前执行程序处理的数据的加密属性；以及

数据加密单元，被进行配置以根据存储在执行状态存储单元的加密属性对将由当前执行程序处理的数据进行加密。

15. 根据权利要求 1 所述的微处理器，该微处理器进一步包括：

执行状态存储单元，用于存储当前执行程序的执行状态、将被当前执行程序处理的数据的加密属性以及用于规定加密属性的加密属性规定信息；

相关信息写入单元，被进行配置以将涉及加密属性规定信息并含有利用秘密密钥获得的签名的相关信息写入外部存储器；

相关信息读出单元，被进行配置以根据将由当前执行程序引用的数据的地址从外部存储器读出相关信息；

数据引用许可单元，被进行配置以利用公开密钥验证包含在相关信息内的签名，并且只有当相关信息内的签名与微处理器的原始签名一致时，才允许当前执行程序根据相关信息和规定信息的加密属性，通过确定密钥和用于数据引用的算法进行数据引用；以及

数据加密单元，被进行配置以根据存储在执行状态存储单元的加密属性将由当前执行程序引用的数据加密。

16. 根据权利要求 1 所述的微处理器，该微处理器进一步包括：

高速缓冲存储器，用于以高速缓存行为单位高速缓存多个程序的明文指令和明文数据，该高速缓冲存储器具有属性区用于各高速缓存行，指明在解密其指令被高速缓存到各高速缓存行的各程序或其执行会将明文数据高速缓存到各高速缓存行的各程序时用于唯一标识解密密钥的解密密钥标识符；

高速缓存访问控制单元，被进行配置以只有当加密属性对一个高速缓存行指明的解密密钥标识符与加密属性对另一个高速缓存行指明的解密密钥标识符一致时，允许通过根据另一个高速缓存行内的一个高速缓存数据执行一个存储在一个高速缓存行的高速缓存程序引起的数据引用。

17. 根据权利要求 16 所述的微处理器，其中当不允许进行数据引用时，将新数据从外部存储器高速缓存到另一个高速缓存行。

18. 根据权利要求 16 所述的微处理器，其中当不允许进行数据引用时，保护异常中断此高速缓存程序的执行。

19. 根据权利要求 1 所述的微处理器，其中执行单元还执行明文程序，并具有调试功能块，在明文程序的执行期间，当执行特定地址或地址区域的程序时或将数据引用到特定地址或地址区域的数据时，该调试功能块用于产生异常，在执行加密程序期间，此调试程序无效。

20. 根据权利要求 1 所述的微处理器，其中该微处理器的各组成单元包含在单一芯片或单一封装内。

01.02.14

说　　明　　书

抗干预微处理器

本发明涉及可以在多任务程序执行环境下防止非法变更执行码和非法处理目标数据的微处理器。

最近几年，微处理器的性能得到显著改善，以致微处理器除了具有传统的诸如计算和图形功能外，还可以实现对视频图像和音频声音的再生和编辑。通过在为最终用户设计的系统（以下简称：PC）中实现这种微处理器，用户可以在监视器上欣赏各种视频图像和音频声音。此外，通过将 PC 的再生视频图像和音频声音的功能与计算能力相结合，可以改善对游戏等的适用性。这种微处理器不是专为某种特定硬件设计的而是可以在各种硬件中实现，因此其优势在于，通过简单更换执行程序的微处理器，已经拥有 PC 的用户花费不多就可以欣赏视频图像和音频声音的再生和编辑。

如果在 PC 上处理视频图像和音频声音，就会产生原始图像和音乐的版权保护问题。在 MD 或数字视频重放装置中，通过在这些装置中事先实现防止非法复制的机制，可以防止无限复制。虽然这种装置还在制造，但是试图通过拆除或改变这些装置来进行非法复制的情况却很少，而且世界范围内的趋势是通过法律禁止制造和销售为了进行非法复制能够改变的装置。因此，由于基于硬件进行非法复制造成的损害并不很严重。

然而，在 PC 上对图像数据和音乐数据进行处理是通过软件进行的而不是通过硬件进行的，并且最终用户可以在 PC 上随意改变软件。即，如果用户具有某些知识，则通过分析程序并重写可执行软件，可以非常容易地进行非法复制。此外，不同于硬件的问题是，这样产生的用于非法复制的软件可以通过诸如网络的各种媒体迅速传播。

为了解决这些问题，用于再生诸如商业电影或音乐的版权保护内容的 PC 软件，传统上采用一种通过对软件进行加密防止分析和变更

01.02.14

的技术。这种技术就是抗干预软件（参考 David Aucsmith 等人在 Proceedings of the 1996 Intel Software Developer's Conference 上发表的 "Tamper Resistant Software: An Implementation"）。

在防止通过 PC 向用户提供的有价值信息（不仅包括视频数据和音频数据而且包括文本和技术诀窍）的非法复制方面，以及在防止 PC 软件本身的技术诀窍被分析方面，抗干预软件技术仍然有效。

然而，抗干预软件技术是一种，通过在开始执行程序之前对要求保护的程序的一部分进行加密，在执行该部分之前对该部分立即进行解密并在该部分执行完毕后立即对该部分再加密，使得难于利用诸如反汇编程序或调试程序的软件工具进行分析。因此，只要处理器可以执行该程序，通过从程序的启动处开始一步一步进行分析总可以对程序进行分析。

此事实成为版权所有人向系统提供版权保护内容用于利用 PC 再生视频数据和音频数据的障碍。

在这方面，其它抗干预软件应用程序也易受攻击，并且此事实成为通过 PC 进行高级信息服务和将含有企业或个人技术诀窍的程序应用到 PC 的障碍。

总之，在软件保护方面同样存在这些问题，此外，PC 是开放式平台，因此存在通过变更被确定为系统软件配置基础的操作系统（OS）进行攻击问题。换句话说，通过使用属于 OS 的特权，怀有恶意的熟练用户可以变更其自有 PC 的 OS 来废除或分析插入到应用程序内的版权保护机制。

当前的 OS 通过利用对存储器的特权操作功能和 CPU 中提供的特权执行控制功能，在计算机的控制下进行资源管理和资源使用仲裁。管理的目标包括传统目标（例如：设备、CPU 和存储资源）以及网络层或应用层 QoS（服务质量）。尽管如此，资源管理的基础仍然是对执行程序所需的资源进行配置。换句话说，分配 CPU 时间来执行此程序并将分配执行程序所需的存储空间是资源管理的基础。通过控制实现访问这些资源的程序的执行（通过分配 CPU 的时间和存储空间），对

01-02-14

其它设备、网络和应用层服务质量 Qos 进行控制.

OS 具有执行 CPU 时间分配和存储空间分配的特权. 换句话说, 为了对 CPU 分配时间, OS 具有在任意时间中断并重新启动应用程序的特权并具有在任意时间将分配到应用程序的存储空间的内容转移到不同分层的存储空间的特权. (通常) 通过利用应用程序的不同访问速度和访问能力隐匿分层存储系统, 将分配到应用程序的存储空间的内容转移到不同分层的存储空间的特权还用于为应用程序提供平面存储器空间.

使用这两种特权, OS 可以中断应用程序的执行状态并在任意时间对它进行快速转储, 并且在对它进行拷贝或重写之后重新启动它. 此功能还可以被用作分析隐藏在应用程序内的秘密的工具.

为了在计算机上防止应用程序被分析, 有几种对程序或数据进行加密的公知技术(例如: Hampson, 第 4,847,902 号美国专利、Hartman, 第 5,224,166 号美国专利、Davis, 第 5,806,706 号美国专利、Takahashi 等, 第 5,825,878 号美国专利、Buer 等人, 第 6,003,117 号美国专利、第 11-282667 号日本公开专利申请 (1999)). 然而, 这些公知的技术均未涉及防止程序运行过程和数据秘密被 OS 进行上述特权操作问题.

基于 Intel 公司开发的 X86 结构的传统技术 (Hartman, 第 5,224,166 号美国专利) 是一种通过利用规定的密钥 Kx 对执行码和数据进行加密以存储执行码和执行数据的技术. 密钥 Kx 可以被表示为 $E_{kp}[Kx]$ 的形式, 利用与嵌入到处理器内的秘密密钥 Ks 对应的公开密钥 Kp, 可以对 $E_{kp}[Kx]$进行加密. 因此, 只有知道 Ks 的处理器可以对存储器上的加密执行码进行解密. 将密钥 Kx 存储到处理器内被称为段式寄存器的寄存器.

利用这种机制, 通过对代码进行加密在某种程度上可以避免用户发现程序代码的秘密. 此外, 对于不知道代码密钥 Kx 的人来说, 由于密码原因难于根据其内涵或利用密钥 Kx 解密时可执行的新产生代码来变更代码.

然而, 采用这种技术的系统的缺点在于, 利用被称为上下文切换

的 OS 特权有可能对程序进行分析，而无需对加密的执行码进行解密。

更具体地说，当利用中断停止执行程序或当预期系统调用程序自行调用软件中断命令时，为了执行其它程序，OS 进行上下文切换。上下文切换操作将指明该点寄存器值的集合的程序执行状态（以下简称为：上下文信息）存储到存储器，并将事先存储到存储器的另一个程序的上下文信息再存入寄存器。

图 15 示出在 x86 处理器中使用的传统上下文存储格式。这里存储了应用程序使用的寄存器的所有内容。当再启动被中断的程序时，将该程序的上下文信息再存入寄存器。为了并行运行多个程序，上下文切换是不可缺少的功能。在传统技术中，在上下文切换时，OS 可以读取寄存器值，因此根据该程序的执行状态是如何改变的，即使不是全部，也可以猜测出该程序执行的大多数操作。

此外，通过控制在此时通过设置计时器等产生异常的时间，在程序的任意执行点可以进行此处理。除了中断执行和分析之外，还可以恶意重写寄存器信息。重写寄存器不仅可以改变程序运行而且可以使得对程序进行分析更容易。OS 可以存储应用程序的任意状态，因此通过重写寄存器值并通过反复运行程序，可以分析程序的运行。除了上述功能之外，处理器还具有诸如逐步执行的调试支持功能，存在的问题是，利用所有这些功能，OS 可以对应用程序进行分析。

就数据而论，第 5, 224,166 号美国专利认为，仅通过利用加密代码段执行程序，程序可以访问加密数据。这里存在的问题是加密程序利用任意密钥可以自由读取加密数据，而与对程序加密的密钥无关，即使存在利用互相不同的密钥加密的程序。这种传统技术中未说明这些情况，即 OS 和应用程序独立具有它们自己的秘密并且应用程序的秘密不被 OS 发现，或者多个程序供应商分别具有它们自己的秘密。

当然，即使是在现有的处理器中，也可以在应用程序之间划分存储空间并利用虚拟存储机制提供的保护功能来禁止应用程序访问系统存储器。然而，只要虚拟存储制受 OS 的控制，则对应用程序秘密的保护就不能依赖于 OS 控制下的功能。这是由于 OS 可以忽略保护机制

访问数据，并且在提供上述虚拟存储器方面，这种特权不可缺少。

作为另一种传统技术，第 11－282667（1999）号日本公开专利申请公开了一种技术，这种技术为了存储应用程序的秘密信息而在 CPU 内设置秘密存储器。在这种技术中，为了访问秘密存储器内的数据，需要规定基准值。但是，此技术未披露如何防止同一个 CPU 内运行的多个程序（特别是 OS）使用用于获得对秘密数据的访问权的基准数值。

此外，在第 5, 123, 045 号美国专利中，Ostrovsky 等人公开了一种系统，该系统的先决条件是使用具有与应用程序对应的唯一秘密密钥的子处理器，在该系统中，不能根据这些子处理器访问主存储器上的程序的访问方式来推测程序运行。这是基于，通过将根据存储器实现运行的指令系统转换到与此指令系统不同的另一个指令系统，实现随机存储访问的机制。

然而，对不同的应用程序，这种技术要求不同的子处理器，因此这种技术的成本高，并且预期用于处理这种指令系统的编译程序和处理器硬件的执行和快速实现过程非常困难，这是由于它们与当前使用的处理器的编译程序和处理器硬件非常不同。除此之外，与上述说明的将程序码和数据简单加密的其它传统技术（例如：第 5, 224,166 号美国专利和第 11－282667 号日本公开专利申请）比较，在这种处理器中，即使当数据和实际操作码的运行被观察到并被跟踪以致调试程序变得非常困难时，难于包含数据内容与运行之间的对应之处，因此，这种技术存在许多实际问题。

因此，本发明的第一个目的是提供一种微处理器，该微处理器即使是在被中断停止执行时也可以防止在多任务环境下内部执行的算法和存储区内的数据状态被非法分析。

此第一个目的受传统技术能够保护程序码的数值而不能防止利用通过发生异常或调试功能中断程序的执行进行分析的启发。因此，本发明的目的是提供一种即使是在程序执行中断时仍能确实保护代码的微处理器，在此微处理器中，这种保护与当前 OS 要求的执行控制功能和存储器管理功能兼容。

01.02.14

本发明的第二个目的是提供一种即使执行多个利用不同密钥加密的程序，其各程序均可以独立获得正确可读/可写数据区的微处理器。

此第二个目的是受第 5, 224,166 号美国专利公开的传统技术的启发，该技术仅提供简单保护，其中禁止利用非加密代码访问加密数据区，并且不可能独立地对多个程序保护它们的秘密。因此，本发明的目的还在于提供一种当多个应用程序具有它们各自的（加密的）秘密时具有用于防止各应用程序的秘密被 OS 使用的数据区的微处理器。

本发明的第三个目的是提供一种可以防止上述数据区的保护属性（即加密属性）被 OS 非法重写的微处理器。

此第三个目的是受第 5, 224, 166 号美国专利公开的传统技术的启发，该技术的缺点在于，通过利用上下文切换中断程序的执行，OS 可以重写在段式寄存器内设置的加密属性。一旦，通过重写加密属性，程序进入以明文形式写入数据的状态，不加密就不将数据写入存储器。即使在某些时间应用程序校验段式寄存器的数值，但是，如果此后重写寄存器的数值，则结果相同。因此，本发明的目的还在于提供一种微处理器，该微处理器具有可以禁止这种变更或可以检测这种变更并可以对这种变更采取适当措施的机制。

本发明的第四个目的是提供一种微处理器，该微处理器可以防止加密属性受密码分析原理的所谓选择明文攻击法攻击，在密码分析原理中，程序可以使用任意数值作为数据密钥。

本发明的第五个目的是提供一种微处理器，该微处理器具有程序调试和反馈的机制。换句话说，本发明目的在于提供一种处理器，在该微处理器中，在执行失败时，可以以明文的形式调试程序并将关于缺陷的反馈信息送到程序码供应商（程序销售商）。

本发明的第六个目的是提供一种微处理器，该微处理器可以以低成本高性能的形式实现上述第一至第五个目的。

为了实现第一个目的，本发明的第一个方面具有下列特征。通过提供读取功能的总线接口单元，制成单芯片或单封装的微处理器从微处理器外部的存储器（例如：主存储器）读取多个利用代码密钥加密

的程序。对于不同的程序，代码密钥不同。利用分别对应的解密密钥，解密单元对这些读出的程序进行解密，并且指令执行单元执行这些已解密程序。

在中断多个程序中一些程序的执行时，提供执行状态写入功能的上下文信息加密/解密单元利用对微处理器唯一的密钥对指明执行状态的信息进行加密直到中断程序的中断点和代码密钥出现，并将加密的信息作为上下文信息写入微处理器外部的存储器。

如果重新启动被中断的程序，提供重新启动功能的验证单元利用与微处理器的唯一密钥对应的唯一解密密钥解密上下文信息，并只有当包含在已解密上下文信息内的代码密钥（即：预定重新启动程序的代码密钥）与已中断程序的原始代码密钥一致时，才重新执行程序。

此外，为了实现第二和第三个目的，微处理器还具有：存储区（例如：寄存器），它在处理器的内部而且不能被读出到外部；加密属性写入单元（例如：指令 TLB），用于将程序的处理目标数据加密属性写入存储器。加密属性包括程序的代码密钥和加密目标地址范围。在上下文信息中至少含有一部分加密属性。

上下文信息加密/解密单元还将对微处理器唯一的、基于秘密信息的签名附加到上下文信息。这样，验证单元判别解密上下文信息内的签名是否与对微处理器唯一的、基于秘密信息的原始签名一致，如果一致，就重新启动已中断的程序。

同样，将加密程序中断点前的执行状态存储到外部存储器作为上下文信息，而将执行处理目标数据的保护属性存储到处理器内部的寄存器，因此，可以防止非法变更数据。

为了实现第四个目的，本发明的第二个方面具有下列特征。制成单芯片或单封装的微处理器在其内保持不能读出到外部的唯一秘密密钥。具有读取功能的总线接口单元事先从微处理器外部的存储器读取利用与秘密密钥对应的、微处理器的唯一公开密钥加密的代码密钥。具有第一解密功能的密钥解密单元利用微处理器的秘密密钥对读出的代码密钥进行解密。总线接口单元还从外部存储器读出多个利用分别

01.02.14

不同的代码密钥加密的程序。具有第二解密功能的代码解密单元对这些读出的程序进行解密。指令执行单元执行执行解密的程序。

如果中断多个程序中一些程序的执行，则随机数发生装置可以产生随机数作为临时密钥。上下文信息加密/解密单元将：第一数值，利用随机数，通过对指明中断程序的执行状态的信息进行加密获得；第二数值，利用中断程序的代码密钥，通过对此随机数进行加密获得；以及第三数值，利用微处理器的秘密密钥，通过对此随机数进行加密获得，写入外部存储器作为上下文信息。

如果重新启动执行程序，上下文信息加密/解密单元从外部存储器读出上下文信息，利用秘密密钥对上下文信息内的第三数值的随机数进行解密，并利用解密的随机数对上下文信息内的执行状态信息进行解密。同时，利用预定重新启动程序的代码密钥，对上下文信息内第二数值的随机数进行解密。将通过利用代码密钥解密第二数值获得的随机数和通过利用秘密密钥解密第三数值获得的随机数与临时密钥进行比较，并且仅在它们一致时，重新启动执行程序。

同样，利用在各存储时刻产生的随机数，将指明中断点时执行状态的上下文信息进行加密，并附加使用对微处理器唯一的秘密密钥的签名，因此，可以将上下文信息安全地存储到外部存储器。

为了实现第一至第三个以及第六个目的，本发明的第三个方面具有下列特征。制成单芯片或单封装的微处理器读出多个利用对不同程序不同的密钥加密的程序并执行它们。此微处理器具有不能读出到外部的内部存储器（例如：寄存器），此微处理器将将由各程序引用的数据（即处理目标数据）的加密属性和说明信息的加密属性存储到寄存器。上下文信息加密/解密单元将相关信息写入外部存储器，此相关信息与存储在寄存器内说明信息的并含有对微处理器唯一的签名的加密属性有关。根据程序提交的数据地址，保护表管理单元从外部存储器读取相关信息。利用秘密密钥，验证单元验证包含在所读出的相关信息内的签名。并且只有当此签名与对微处理器唯一的签名一致时，验证单元才根据说明信息的加密属性和读出的相关信息，允许程序引用

01.02.14

数据。

在这种配置中，待存储到内部寄存器的信息与签名附在一起并存储到外部存储器，并且只将必要部分读出到微处理器。在读取时验证签名，可以确保不受代换之害。即使当增加要处理的程序数并增加加密属性的数目时，也无需扩大微处理器内的存储区，因此可以降低成本。

根据本发明的一个方面，提供了一种微处理器，该微处理器具有与不能读出到外部的唯一秘密密钥对应的唯一秘密密钥和唯一公开密钥，它包括：读取单元，用于从外部存储区读出多个利用不同执行码密钥加密的程序；解密单元，被进行配置以利用各自解密密钥，对多个通过读取单元读出的程序进行解密；执行单元，被进行配置以执行多个通过解密单元解密的程序；上下文信息保存单元，被进行配置以将其执行被中断的程序的上下文信息保存到外部存储器或保存到在微处理器内部设置的上下文信息存储器，该上下文信息含有指明此程序的执行状态的信息和此程序的执行码密钥；以及重新启动单元，被进行配置以通过从外部存储器或上下文信息存储器读出上下文信息并通过从上下文信息中恢复此程序的执行状态，重新启动执行此程序。

通过以下结合附图的描述，本发明的其它特征和优势将会更加明显。

图 1 示出根据本发明第一实施例具有微处理器的系统的方框图。

图 2 示出在图 1 所示的微处理器内使用的全部存储空间的示意图。

图 3 示出根据本发明第二实施例的微处理器的基本配置的方框图。

图 4 示出图 3 所示的微处理器的详细配置的方框图。

图 5 示出在图 3 所示的微处理器中使用的页目录格式和页表格式的示意图。

图 6 示出在图 3 所示的微处理器中使用的页表格式和密钥输入格式。

图 7A 和图 7B 分别示出在图 3 所示的微处理器中使用的、交错前

01.04.19

和交错后的典型数据。

图 8 示出在图 3 所示的微处理器内执行的代码解密过程的信息流.

图 9 示出在图 3 所示的微处理器内使用的 CPU 寄存器.

图 10 示出在图 3 所示的微处理器内使用的上下文保持格式.

图 11 示出在图 3 所示的微处理器内执行的保护域切换过程的流程图.

图 12 示出在图 3 所示的微处理器内执行的数据加密和解密处理过程的信息流.

图 13 示出由图 3 的微处理器执行保护域内控制的处理的流程图.

图 14 示出概念性说明利用图 3 所示的微处理器的调用过程和从保护域转移到非保护域的过程的示意图.

图 15 示出在传统处理器内使用的上下文保存格式.

现在将参考图 1 和图 2 详细说明根据本发明的抗干预微处理器的第一实施例.

此第一实施例涉及一种微处理器，该微处理器用于防止程序指令秘密（执行码）和上下文信息（执行状态）被目标系统的用户使用，其中利用公开密钥（非对称密钥）加密系统以加密形式提供程序指令秘密和上下文信息.

图 1 示出目标系统，通过总线 2102，将目标系统的微处理器 2101 连接到主存储器 2103.

如图 1 所示，在此实施例中，微处理器 2101 具有寄存器文件 2111、指令执行单元 2112、指令缓冲器 2113、公开密钥解密功能块 2114、秘密密钥寄存器 2115、公共密钥解密功能块 2116、公共密钥寄存器 2117、BIU（总线接口单元）2118、缓冲寄存器 2119、公开密钥寄存器 2120、加密功能块 2121、解密功能块 2122 以及以前的公共密钥寄存器 2123，以下将对它们进行详细说明.

首先对将在下述描述中使用的术语进行说明，然后主要说明通用操作系统（OS）和应用程序的操作. 程序是为特定目的编写的一组数据和一系列机器语言指令. OS 是用于管理系统资源的程序，而应用程

序是在 OS 资源管理的管理下运行的程序，该实施例预先支持多任务系统，因此，多个应用程序可以在 OS 的管理下以准并行的方式运行。以准并行方式运行的每个应用程序被称为进程。有时，将为相同目的执行进程的一组进程称为任务。

通常以文件的形式将应用程序的指令和数据存储到二级存储器。利用 OS 的装载程序，将它们设置到存储器，然后将它们作为进程执行。通常，利用输入/输出等引起的处理器异常处理（或中断）中断程序的执行。将执行异常处理的程序称为异常处理程序。通常利用 OS 安装异常处理程序。OS 可以处理硬件的异常请求、中断应用程序的运行过程并在任意时间重新启动或启动另一个应用程序。中断进程包括：无需在执行异常处理程序后切换进程，就重新启动原始进程的执行的暂时情况；以及要求切换进程的情况。前者的例子具有简单计时器而后者的例子具有由于页异常处理的虚拟存储器。

此实施例的目的是防止程序指令（执行码）和执行状态被目标系统用户使用，目标系统用户可以自由读目标系统主存储器并可以自由变更 OS 程序或应用程序。

实现此目的的基本特征是对处理器内信息存储的访问控制和根据下列信息的加密过程。

（1） 程序创建者选择的公共密钥 Kx，利用使用此密钥的秘密密钥加密系统对应用程序进行加密。

（2） 一对在处理器内设置的唯一公开密钥 Kp 和唯一秘密密钥 Ks。利用指令程序可以读出公开密钥。

（3） 密钥信息，其中的程序公共密钥 Kx 被利用处理器的公开密钥 Kp 加密。

[明文程序的执行]

此处理器可以执行具有共存明文指令和加密指令并被设置到主存储器的程序。这里将参考图 1 和图 2 所示的存储器分配，对在 CPU 内执行明文程序的运行过程进行说明。

图 2 示出整个存储器空间 2201，在存储器空间中，将程序设置到

01.02.14

主存储器上的区域 2202 至 2204，其中区域 2202 和 2204 为明文区域，而区域 2203 为加密区域。区域 2205 存储解密区域 2203 时使用的密钥信息。

当利用转移指令将控制从 OS 跳转到程序等的顶部 x 时，启动该程序的执行。指令执行单元 2112 执行跳转到 x 的指令，并将指令的地址输出到 BIU 2118。通过总线 2102 读取地址 x 的内容，并将地址 x 的内容从 BIU 2118 送到指令缓冲器 2113，然后送到指令执行单元 2112 执行该指令。其运行结果反应到寄存器文件 2111。当运行目标是对主存储器 2103 上的地址的读/写时，将其地址数值送到 BIU 2118，将此地址从 BIU 2118 输出到总线 2102，然后对存储器进行读/写操作。

指令缓冲器 2113 具有存储两条或多条指令的容量，并且从主存储器 2103 选择性地读出与指令缓冲器 2103 的容量对应的指令。

[加密指令的执行]

接着说明加密指令的执行情况。根据此实施例的处理器具有两种状态：明文指令的执行状态和加密指令的执行状态。因此提供了两种指令用于控制这两种状态。一种指令是加密执行启动指令，用于实现从明文指令的执行状态转换到加密指令的执行状态；另一个指令是明文返回指令，用于实现相反的转换。

[加密执行启动指令]

加密执行启动指令被表示为如下的助记符号“execenc”并具有一个操作数：

execenc keyaddr

其中“keyaddr”表示存储解密后续指令时使用的密钥信息的地址。

[密钥信息]

在此将说明密钥信息和程序加密过程。加密区域 2203 包括加密指令序列。将指令细分为以预取指令队列大小为单位的块，并利用诸如 DES（数据加密标准）算法的秘密密钥算法对指令进行加密。以下将此加密过程中使用的密钥表示为 Kx。由于使用了秘密密钥算法，所以解密时使用相同的密钥。

01.02.14

如果以明文的形式将此 Kx 设置到主存储器，则可以控制 OS 的用户可以容易地读取它并对加密程序进行分析。 为了防止这种情况发生，将通过利用处理器的公开密钥 Kp 加密 Kx 获得的 $E_{k\ p}$[Kx]设置到存储器区域 2205. "keyaddr" 表示区域 2205 的顶地址。

除非知道与公开密钥 Kp 对应的 Ks，否则不可能用密码方法（计算方法）对 $E_{k\ p}$[Kx]进行解密获得 Kx. 因此，只要目标系统的用户不知道 Ks，就一定不会将程序的秘密泄露给用户。在处理器内部，以不能由外部读取的方式存储此 Ks. 处理器可以在内部解密 Kx，而不会使用户得知此 Ks，而且处理器还可以利用 Kx 对加密程序进行解密并执行此程序。

以下将详细说明加密执行启动指令和后续的加密指令的执行。通过执行区域 2207 内的转移指令，控制被转移到地址 "启动" 处的加密执行启动指令。在加密执行启动指令的操作数 "keyaddr" 指明的地址，将规定区域 2205 的内容作为数据读出到处理器的指令执行单元 2112. 指令执行单元 2112 将此数据 $E_{k\ p}$[Kx]送到公开密钥解密功能块 2114. 通过利用在秘密密钥寄存器 2115 存储器储的、对处理器唯一的秘密密钥 Ks 解密 $E_{k\ p}$[Kx]获得 Kx. 并将它存储到公共密钥寄存器 2117. 然后，处理器进入加密指令执行状态。

在此，假设这样制造处理器封装，以致不能利用处理器芯片的程序和调试程序将秘密密钥寄存器 2115 和公共密钥寄存器 2117 存储器储的内容读出到外部。

通过执行加密执行启动指令，将解密后续指令使用的密钥存储到公共密钥寄存器 2117，然后处理器进入加密指令执行状态。当处理器处于加密指令执行状态时，将从主存储器 2103 读取的指令由 BIU 2118 送到公共密钥解密功能块 2116，并利用存储在公共密钥寄存器 2117 内的密钥信息对它进行解密，然后将它存储到指令缓冲器 2113.

在此实施例中，紧跟在加密执行启动指令之后存储到区域 2204、利用密钥 Kx 加密的程序被解密并被存储到指令缓冲器 2113，然后执行它。以指令缓冲器 2113 的大小为单位进行读取。图 2 示出指令缓冲

01-02-14

器 2113 的大小为 64 位的典型情况，并且选择性地将 16 位大小的四条指令分别读出到指令缓冲器 2113.

[明文返回指令]

通过执行明文返回指令，处于加密指令执行状态的处理器返回明文指令执行状态。

明文返回指令被表示为如下助记符号:

exitenc

它没有操作数. 通过执行此指令，经过不通过公共密钥解密功能块 2116 的通路从主存储器 2103 读取该指令，然后处理器返回明文指令执行.

请注意，当在加密指令执行期间再执行加密执行启动指令时，改变指令解密密钥，以致可以利用不同密钥解密后续指令，然后执行后续指令.

[上下文保存和对其的攻击]

接着将说明为了在多任务环境下保护应用程序的秘密而安全保存执行状态的情况.

此处理器的寄存器文件 2111 具有 32 个通用寄存器（R0 至 R31）. R31 用作程序计数器. 将通用寄存器的内容存储到寄存器文件 2111. 当在上述加密程序的执行期间发生异常时，将寄存器文件 2111 的内容转移到缓冲寄存器 2119, 并用预定数值或随机数初始化寄存器文件 2111 的内容. 然后，将用于解密加密程序的公共密钥数值存储到前一个公共密钥寄存器 2123. 只有完成这两种初始化之后，才可以将控制转移到异常处理程序并执行异常处理程序的指令. 假设异常处理程序的指令未加密.

利用此寄存器文件的初始化功能，在此实施例的处理器中，即使是在由于加密程序的执行期间发生异常而将控制转移到异常处理程序的情况下，仍可以防止读取加密程序利用异常处理程序处理的寄存器数值. 同时，将寄存器文件 2111 的内容保存到缓冲寄存器 2119. 以下将说明为了重新启动加密程序，用于恢复缓冲寄存器内容并用于将它

们存储到存储器的功能块.

可以直接由异常处理程序的非加密程序读出存储在缓冲寄存器2119 的寄存器内容. 异常处理程序的非加密程序只允许对缓冲寄存器2119进行如下两步操作.

（1） 恢复缓冲寄存器内容并重新启动执行原始加密程序.

（2） 加密缓冲寄存器的内容并将它们存储到存储器，然后执行OS 程序或其它加密程序.

在（1）操作情况下，当处理诸如计数器递增的异常处理程序完毕后，异常处理程序发出“cont”（继续）指令. 当执行“cont”指令时，在寄存器文件2111和公共密钥寄存器2117内分别恢复缓冲寄存器2119的内容和前一个公共密钥寄存器2123的内容. 由于在寄存器文件2111内含有程序计数器，因此通过使控制退回到中断加密程序执行的点，可以重新启动加密程序的执行. 为了在重新启动之后对加密程序解密，可以使用从前一个公共密钥寄存器 2123 恢复的数值. 与缓冲寄存器2119的内容相同，显然，该程序不能重写前一个公共密钥寄存器2123.

（2）操作情况与在执行异常处理程序定时发生的进程切换的情况对应. 在这种情况下，处理器的异常处理程序或任务调度程序发出“savereg”（保存寄存器）指令用于将缓冲寄存器 2119 的内容保存到存储器. 此“savereg”指令被表示为如下助记符号:

savereg dest

并且该指令具有一个操作数“dest”，操作数“dest”代表保存缓冲寄存器内容的地址.

发出“savereg”指令时，通过使用存储在公开密钥寄存器 2120内的处理器公开密钥 Kp，利用加密功能块2121对缓冲寄存器2119和前一个公共密钥寄存器 2123 的内容进行加密，并通过 BIU2118 将它们保存到主存储器 2103 内由“dest”规定的地址. 主存储器 2103 在处理器的外部，因此有被用户访问的可能性，但是利用处理器的公开

密钥可以将这些内容加密，这样不知道处理器的秘密密钥的用户不可能得知缓冲寄存器的内容。

保存缓冲寄存器的内容后，利用上述方法，OS 激活另一个加密程序。如果未保存缓冲寄存器的内容就激活另一个加密程序，则当中断另一个加密程序的执行时，会将缓冲寄存器的内容重写到另一个加密程序的缓冲寄存器，并且由于原始加密程序已丢失，所以不可能将原始加密程序作为缓冲寄存器内容重新启动。

在此，假定缓冲寄存器的个数为 1，但是为了处理多个异常，也可以具有多个缓冲寄存器。

[恢复过程]

接着，将说明已保存执行状态的恢复过程。

在重新启动被中断应用程序时，OS 的调度程序发出“rcvrreg”（恢复寄存器）指令。此“rcvrreg”指令被表示为如下助记符号:

rcvrrdg addr

并且该指令具有一个操作数“addr”，该操作数“addr”代表保存执行状态的地址。

发出“rcvrreg”指令指令时，利用处理器 BIU 2118，从“addr”规定的存储器地址取出加密执行状态信息，利用解密功能块 2122，通过使用处理器的秘密密钥 Ks 对它进行解密，然后在寄存器文件 2111 内恢复寄存器信息，而在公共密钥寄存器 2117 内恢复程序解密密钥。恢复完成时，从程序计数器指明的点重新启动已中断程序的执行。此时，将从执行状态信息中恢复的密钥 Kx 用于加密程序的解密过程。

以上说明了由于异常中断的加密程序的执行状态的保存过程和恢复过程的细节。如上所述，加密程序可以避免受到可以控制目标系统 OS 用户的攻击。

接着，将说明防止两种攻击执行状态的方式的上述方法的安全性。

[攻击执行状态]

有两种方式可以攻击应用程序执行中产生的执行状态。一种方式是利用攻击者窥视保存的执行状态，而另一种方法是利用攻击将执行状态重写为要求数值。

在此，将定义如下两个用于解释非法访问执行状态的术语。首先，将产生执行状态的程序称为此执行状态的原始程序。通过在寄存器内恢复执行状态，可以重新启动原始程序。另外，将产生执行状态的程序以外的程序（即利用不同于原始程序的密钥或明文程序的密钥的密钥加密的程序）称为其它程序。

将对某些原始程序产生的执行状态的非法访问或攻击定义为，不知道原始程序密钥的第三方利用某些独立于处理器操作的方法对存储器上的执行状态进行直接分析的行为，或第三方利用在相同处理器上运行的其它程序分析执行状态或将执行状态重写为要求的数值的行为。

在此实施例的微处理器中，利用如下三种机制可以保护执行状态，这样就可以防止利用访问处理器外的存储器或利用其它程序进行非法访问。

首先，在此实施例中，当加密程序的执行被中断时，将寄存器信息保存到缓冲寄存器 2119。然后，利用“rcvrreg”指令或“savereg”指令方法之外的任何方法均不能访问缓冲寄存器 2119 和前一个公共密钥寄存器 2123，所以，其它程序不能自由读取它们的内容。

在传统处理器中，利用异常处理程序可以自由读取发生异常时的寄存器内容。在此实施例的微处理器中，寄存器内容被保存到缓冲寄存器 2119，因此可以禁止其它程序读取寄存器内容，为了防止系统用户窥视存储在存储器上的执行状态，提供用于通过利用处理器的公开密钥对它们进行加密来保存缓冲寄存器内容的指令。

第二种攻击方法是，通过在与原始程序相同的存储器地址设置为攻击者所知的某些其它程序指令来读取包含在执行状态内的寄存器数值，以致使此其它程序读取加密执行状态。

在此实施例的微处理器中，加密执行状态含有程序密钥，并且在

重新启动时将此密钥用于解密加密程序。由于有此机制，即使在原始程序之外的其它程序试图读取执行状态时，由于密钥不匹，所以不能将程序直接解密并且不能按照攻击者的意图执行程序。这样，在此实施例的微处理器中就不可能使用第二种攻击方法。

通过利用处理器的公开密钥简单解密执行状态本身不能实现此效果，但是通过对原始程序的密钥和执行状态整体进行加密可以实现此效果。

请注意，为了使此效果最好，在使用公开密钥进行加密时，优先将寄存器（R0 至 R31）的数值和公共密钥 Kx 存储到相同的密码块。

[数据保护]

在此实施例的微处理器中，未考虑对数据进行加密。但是，对于本技术领域的技术人员来说显而易见，与在微处理器中进行数据加密用以支持将在第二实施例中说明的虚拟存储器相同，可以将数据加密功能添加到此实施例的微处理器。

现在参考图 3 至图 14，详细说明根据本发明抗干预微处理器的第二实施例。

在此实施例中，对于使用基于 Intel 公司推出的、广泛使用的 Pentium Pro 微处理器结构的典型情况，说明根据本发明的微处理器，但是本发明并不局限于此特定结构。在以下的说明中，将说明 Pentium Pro 微处理器结构的具体特征并将说明使用其它结构的情况。

请注意，Pentium Pro 结构对地址空间划分了三种地址，它们包括：物理地址、线性地址以及逻辑地址，但是在此实施例中将 Pentium 术语中的线性地址称为逻辑地址。

在以下的说明中，除非另有说明，保护包括对应用程序秘密的保护（即利用加密进行保护）。因此，应清楚地将此实施例中的保护与通常使用的保护概念区别开，这是预防由于某些程序的运行对其它应用程序运行的干扰。然而，在本发明中，在普通意义上，当然由 OS 提供运行保护机制（由于它与本发明无关，所以省略了对这方面的说明），该保护机制与根据本发明应用程序的秘密保护并行。

01.02.14

此外，在以下说明中，将处理器可以执行的机器语言指令称为指令，并且选择性地将多条指令称为执行码或指令流。将加密指令流过程使用的密钥称为执行码密钥。

此外，在以下说明中，将秘密保护机制称为在 OS 管理下的应用程序保护秘密，但是可以将此机制用作防止 OS 本身被变更或分析的机制。

图 3 示出根据此实施例的微处理器的基本配置。图 4 示出图 3 所示的微处理器的详细配置。

微处理器 101 具有：处理器核心 111、指令 TLB（查表缓冲器）121、异常处理单元 131、数据 TLB（查表缓冲器）141、二级高速缓存 152。处理器核心 111 包括总线接口单元 112、代码和数据加密/解密处理单元 113、一级高速缓存 114 以及指令执行单元 115。指令执行单元 115 进一步包括指令提取/解码单元 214、指令表 215、指令执行转换单元 216 以及指令执行完成单元 217。

异常处理单元 131 进一步包括寄存器文件 253、上下文信息加密/解密单元 254、异常处理单元 255、秘密保护破坏检测单元 256 以及执行码密钥与签名验证单元 257。

指令 TLB 121 进一步包括页表缓冲器 230、执行码解密密钥表缓冲器 231 以及密钥单元 232。数据 TLB 141 进一步包括保护表管理单元 233。

微处理器 101 具有用于存储对此微处理器唯一的公开密钥 Kp 和秘密密钥 Ks 的密钥存储区 241。现在研究从某些程序销售商购买要求的执行程序 A 并执行它的情况。程序销售商在提供执行程序 A 之前利用公共执行码密钥 Kcode（$E_{Kcode}[A]$）将程序 A 加密，然后将用于以利用微处理器 101（$E_{Kp}[Kcode]$）公开密钥 Kp 的加密方式进行加密的公共密钥 Kcode 送到微处理器 101。微处理器 101 是一种多任务处理器，它不仅可以处理执行程序 A 而且可以以准并行方式处理多个不同的加密程序（即通过允许中断实现）。此外，微处理器 101 可以预先执行加密程序和明文程序。

通过总线接口单元（读取功能块）112，微处理器 101 从微处理器 101 外部的主存储器 281 读出多个利用不同的执行码密钥加密的程序。利用各自对应的解密密钥，执行码解密单元 212 对此多个读出程序进行解密，然后，指令执行单元 115 执行此多个解密程序。

在中断一些程序的执行的情况下，利用微处理器的公开密钥，异常处理单元 131 的上下文信息加密/解密单元 254 将指明被中断程序中断点处执行状态和此程序的代码密钥的信息加密，然后将此加密信息作为上下文信息写入主存储器 281.

在重新启动中断程序的情况下，利用微处理器 101 的秘密密钥，执行码密钥与签名验证单元 257 将加密上下文信息解密，验证解密上下文信息内的执行码密钥（即：预定重新启动程序的执行码密钥）是否与中断程序的原始执行码密钥一致，只有在它们一致时才重新启动该程序的执行。

在说明微处理器 101 的详细配置和功能之前，这里先概括说明利用微处理器 101 对明文指令执行和执行加密程序的处理过程。

当微处理器 101 执行明文指令时，指令提取/解码单元 214 试图从 L1 指令高速缓存 213 读取由程序计数器（未示出）指明的地址上的内容。如果规定地址上的内容被高速缓存，则从 L1 指令高速缓存 213 读出程序并将此程序送到指令表 215，然后执行它。指令表 215 可以并行执行多条指令，请求将完成执行的必要数据读到指令执行转换单元 216，然后接收此数据。当并行执行指令并且确定它们的执行结果时，将执行结果送到指令执行完成单元 217. 当运行目标是微处理器 101 内的寄存器时，指令执行完成单元 217 将执行结果写入寄存器文件 253；当执行目标是存储器时，指令执行完成单元 217 将执行结果写入 L1 数据高速缓存 218.

L1 数据高速缓存 218 的内容在总线接口单元 112 的控制下被 L2 高速缓存 152 再一次高速缓存，并被写入主存储器 281. 这里使用了虚拟存储器机制，图 5 所示的页表说明逻辑存储地址与物理存储地址有相似之处。

01-02-14

页表是一种设置到物理存储器的数据结构. 实际上, 数据 TLB141 实现从逻辑地址到物理地址的转换, 同时管理数据高速缓存. 根据微